Classes & Objects in Detail:

So far, you have got very basic idea about C++ Classes and Objects. There are further interesting concepts related to C++ Classes and Objects which we will discuss in various sub-sections listed below:

|  |  |
| --- | --- |
| **Concept** | **Description** |
| [Class member functions](http://www.tutorialspoint.com/cplusplus/cpp_class_member_functions.htm) | A member function of a class is a function that has its definition or its prototype within the class definition like any other variable. |
| [Class access modifiers](http://www.tutorialspoint.com/cplusplus/cpp_class_access_modifiers.htm) | A class member can be defined as public, private or protected. By default members would be assumed as private. |
| [Constructor & destructor](http://www.tutorialspoint.com/cplusplus/cpp_constructor_destructor.htm) | A class constructor is a special function in a class that is called when a new object of the class is created. A destructor is also a special function which is called when created object is deleted. |
| [C++ copy constructor](http://www.tutorialspoint.com/cplusplus/cpp_copy_constructor.htm) | The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously. |
| [C++ friend functions](http://www.tutorialspoint.com/cplusplus/cpp_friend_functions.htm) | A **friend** function is permitted full access to private and protected members of a class. |
| [C++ inline functions](http://www.tutorialspoint.com/cplusplus/cpp_inline_functions.htm) | With an inline function, the compiler tries to expand the code in the body of the function in place of a call to the function. |
| [The this pointer in C++](http://www.tutorialspoint.com/cplusplus/cpp_this_pointer.htm) | Every object has a special pointer **this** which points to the object itself. |
| [Pointer to C++ classes](http://www.tutorialspoint.com/cplusplus/cpp_pointer_to_class.htm) | A pointer to a class is done exactly the same way a pointer to a structure is. In fact a class is really just a structure with functions in it. |
| [Static members of a class](http://www.tutorialspoint.com/cplusplus/cpp_static_members.htm) | Both data members and function members of a class can be declared as static. |

<http://www.sabaq.pk/index.php>

[Aggregation vs Composition](http://programmers.stackexchange.com/questions/61376/aggregation-vs-composition)

|  |  |  |  |
| --- | --- | --- | --- |
|  | I understand what composition is in OOP, but I am not able to get a clear idea of what Aggregation is. Can someone explain? | | |
|  | | | 1. A "owns" B = Composition : B has no meaning or purpose in the system without A 2. A "uses" B = Aggregation : B exists independently (conceptually) from A   Example 1:  A Company is an aggregation of People. A Company is a composition of Accounts. When a Company ceases to do business its Accounts cease to exist but its People continue to exist.  Example 2: (very simplified)  A Text Editor owns a Buffer (composition). A Text Editor uses a File (aggregation). When the Text Editor is closed, the Buffer is destroyed but the File itself is not destroyed. |
|  | | | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | **7** |  | | Last sentence should be: A Text Editor owns a Buffer(composition). A Text Editor uses a File (aggregation). When you apply this correction, I will change my stance on your post from negative (confusing) to positive. :) –  [Ed James](http://programmers.stackexchange.com/users/13125/ed-james) [Mar 25 '11 at 8:37](http://programmers.stackexchange.com/questions/61376/aggregation-vs-composition#comment108397_61527) | | |  |  | | --- | --- | | **3** |  | | So is a car an aggregate or a composition of its parts? –  [reinierpost](http://programmers.stackexchange.com/users/10094/reinierpost" \o "458 reputation) [Mar 31 '11 at 12:13](http://programmers.stackexchange.com/questions/61376/aggregation-vs-composition#comment112110_61527) | | |  |  | | --- | --- | | **1** |  | | And how is aggregation different from any other relationship between two sorts of entities? –  [reinierpost](http://programmers.stackexchange.com/users/10094/reinierpost" \o "458 reputation)[Mar 31 '11 at 12:14](http://programmers.stackexchange.com/questions/61376/aggregation-vs-composition#comment112113_61527) | | |  |  | | --- | --- | | **14** |  | | @reinierpost In *reality*, a car is an aggregation of parts, and parts are simply an aggregation of molecules... However, in a *model* it all depends on your requirements. Is it important to treat the engine as a separate entity so that you can track its lifetime independent of the car? Can you reuse the exact same engine in another car? If so, then you probably want aggregation. Otherwise you want a composition because you don't care about engines that aren't part of cars, nor do you care about reusing engines. –  [Curtis Batt](http://programmers.stackexchange.com/users/13078/curtis-batt) [Mar 31 '11 at 14:06](http://programmers.stackexchange.com/questions/61376/aggregation-vs-composition#comment112177_61527) | | |  |  | | --- | --- | | **1** |  | | what is missing is an implementation example for a complete understanding... –  [Chesnokov Yuriy](http://programmers.stackexchange.com/users/106096/chesnokov-yuriy" \o "101 reputation) [Oct 26 '13 at 10:04](http://programmers.stackexchange.com/questions/61376/aggregation-vs-composition#comment426843_61527) |   [show **2** more comments](http://programmers.stackexchange.com/questions/61376/aggregation-vs-composition) |
|  | | From <http://en.wikipedia.org/wiki/Object_composition>  Aggregation differs from ordinary composition in that it does not imply ownership. In composition, when the owning object is destroyed, so are the contained objects. In aggregation, this is not necessarily true. For example, a university owns various departments (e.g., chemistry), and each department has a number of professors. If the university closes, the departments will no longer exist, but the professors in those departments will continue to exist. Therefore, a University can be seen as a composition of departments, whereas departments have an aggregation of professors. In addition, a Professor could work in more than one department, but a department could not be part of more than one university.  So - while you have an ownership relationship with composition the owned object is also destroyed when the owner is - an aggregation (and the objects contained) can exist independently.  --  Update: Apologies - this answer is far too simplistic in hindsight.  c.batt provides an excellent definition in his answer: [Aggregation vs Composition](http://programmers.stackexchange.com/questions/61376/aggregation-vs-composition/61527#61527)   |  |  |  | | --- | --- | --- | | [share](http://programmers.stackexchange.com/a/61378)[improve this answer](http://programmers.stackexchange.com/posts/61378/edit) | [edited Mar 24 '11 at 22:05](http://programmers.stackexchange.com/posts/61378/revisions) | community wiki  [3 revs, 2 users 94%](http://programmers.stackexchange.com/posts/61378/revisions) [HorusKol](http://programmers.stackexchange.com/users/13421) | | |
|  | | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | **3** |  | | In the example you quote the composition is a one-to-many and the aggregation also has a one-to-many relationship implied, though here it could also be a many-to-many relationship for the aggregation (we can suppose possible that a teacher can teach in multiple departments). Whereas a department cannot be part of multiple universities. Composition implies ownership whereas aggregation does not go beyond relationship. The quote is correct but the comment is not. –  [Newtopian](http://programmers.stackexchange.com/users/13899/newtopian" \o "4362 reputation) [Mar 24 '11 at 5:20](http://programmers.stackexchange.com/questions/61376/aggregation-vs-composition#comment107631_61378) | | |  |  | | --- | --- | | **1** |  | | it has nothing to do with destruction! UML does not define garbage collection system. –  [Display Name](http://programmers.stackexchange.com/users/5260/display-name) [Mar 24 '11 at 10:01](http://programmers.stackexchange.com/questions/61376/aggregation-vs-composition#comment107708_61378) | | |  |  | | --- | --- | | **1** |  | | i think the wikipedia link is getting reflexive upvotes, but this is a terrible definition - as @bold pointed out these relationships have nothing to do with GC. This also falls apart when an object is the component of two other objects, such as the ball in a ball-joint joining two artificial limbs. The Component relationship is about functional dependence. –  [Steven A. Lowe](http://programmers.stackexchange.com/users/906/steven-a-lowe) [Mar 24 '11 at 14:13](http://programmers.stackexchange.com/questions/61376/aggregation-vs-composition#comment107811_61378) | | |  |  | | --- | --- | | **1** |  | | I agree that my answer is severely lacking - but so is the WikiPedia article... –  [HorusKol](http://programmers.stackexchange.com/users/13421/horuskol" \o "2209 reputation) [Mar 24 '11 at 22:13](http://programmers.stackexchange.com/questions/61376/aggregation-vs-composition#comment108125_61378) |   add a comment | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote**16**down vote | | There is no single explanation. Different authors mean different things by aggregation. Most don't really mean anything specific by it.   |  |  | | --- | --- | | [share](http://programmers.stackexchange.com/a/61446)[improve this answer](http://programmers.stackexchange.com/posts/61446/edit) | answered Mar 24 '11 at 9:58  [[https://www.gravatar.com/avatar/e8cbc0b9ed29b027a127d8df309af5a6?s=32&d=identicon&r=PG](http://programmers.stackexchange.com/users/10094/reinierpost)](http://programmers.stackexchange.com/users/10094/reinierpost)  [reinierpost](http://programmers.stackexchange.com/users/10094/reinierpost) **458**47 | |
|  | | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | **3** |  | | This is the correct answer. I've read it in two books, one of them being Martin Fowler's *UML Distilled*. – [davidhaskins](http://programmers.stackexchange.com/users/17612/davidhaskins" \o "1998 reputation) [Mar 25 '11 at 12:52](http://programmers.stackexchange.com/questions/61376/aggregation-vs-composition#comment108483_61446) |   add a comment |
| up vote**8**down vote | * Composition is an Association * Aggregation is an Association * Composition is a *strong* Association (If the life of contained object totally depends on the container object, it is called strong association) * Aggregation is a *weak* Association (If the life of contained object doesn't depends on the container object, it is called weak association)   Example:  class Contained {  public void disp() {  System.out.println("disp() of Contained A");  }  }  public class Container {  private Contained c;  //Composition  Container() {  c = new Contained();  }  //Association  public Contained getC() {  return c;  }  public void setC(Contained c) {  this.c = c;  }  public static void main(String[] args) {  Container container = new Container();  Contained contained = new Contained();  container.setC(contained);  }  }   |  |  |  | | --- | --- | --- | | [share](http://programmers.stackexchange.com/a/204137)[improve this answer](http://programmers.stackexchange.com/posts/204137/edit) | [edited Jul 9 '13 at 8:33](http://programmers.stackexchange.com/posts/204137/revisions)  [[https://www.gravatar.com/avatar/fbf2421aadfca785f0c591afe75197f1?s=32&d=identicon&r=PG](http://programmers.stackexchange.com/users/7422/kilian-foth)](http://programmers.stackexchange.com/users/7422/kilian-foth)  [Kilian Foth](http://programmers.stackexchange.com/users/7422/kilian-foth) **42.5k**10107150 | answered Jul 9 '13 at 7:24  [[https://www.gravatar.com/avatar/bcc5a43397880064d50a328ee5ae2d6b?s=32&d=identicon&r=PG](http://programmers.stackexchange.com/users/96095/pawan)](http://programmers.stackexchange.com/users/96095/pawan)  [Pawan](http://programmers.stackexchange.com/users/96095/pawan) **81**12 | | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | **1** |  | | What is the difference between aggregation and association that is neither composition nor aggregation? – [reinierpost](http://programmers.stackexchange.com/users/10094/reinierpost" \o "458 reputation) [Mar 24 '14 at 14:54](http://programmers.stackexchange.com/questions/61376/aggregation-vs-composition#comment466290_204137) |   add a comment | |

|  |  |  |  |
| --- | --- | --- | --- |
| up vote**3**down vote | aggregation is a simple collection, like a bag of marbles  composition implies internal/functional dependencies, like the hinges on a box  cars aggregate passengers; they get in and out without breaking the car's functionality  the tires are components; remove one and the car no longer functions correctly  [note: the spare tire is an aggregate!]   |  |  | | --- | --- | | [share](http://programmers.stackexchange.com/a/61516)[improve this answer](http://programmers.stackexchange.com/posts/61516/edit) | answered Mar 24 '11 at 14:09  [[https://www.gravatar.com/avatar/4c263fe87d408564a9e5f6d1e519aed4?s=32&d=identicon&r=PG](http://programmers.stackexchange.com/users/906/steven-a-lowe)](http://programmers.stackexchange.com/users/906/steven-a-lowe)  [Steven A. Lowe](http://programmers.stackexchange.com/users/906/steven-a-lowe) **30.8k**63131 | |
|  | add a comment |
| up vote**1**down vote | I always look at composition as 'needs a', i.e. a car *needs an* engine, and I look at aggregation as 'things related for a purpose'. So staying with the car analogy, my aggregation may be to represent a journey which may involve bringing a car and passengers together. The journey does not own the car or the passengers, I'm aggregating data that is related for a specific scenario. When the journey is completed the car and the passengers go on. When a car is ended, the car and it's engine are normally destroyed together.   |  |  | | --- | --- | | [share](http://programmers.stackexchange.com/a/61444)[improve this answer](http://programmers.stackexchange.com/posts/61444/edit) | answered Mar 24 '11 at 9:52  [[https://www.gravatar.com/avatar/941268b0843b1d0b23bd0d2f499c6b89?s=32&d=identicon&r=PG](http://programmers.stackexchange.com/users/2643/lazarus)](http://programmers.stackexchange.com/users/2643/lazarus)  [Lazarus](http://programmers.stackexchange.com/users/2643/lazarus) **371**26 | |
|  | add a comment |

|  |  |
| --- | --- |
| up vote**-1**down vote | Semantically, all sets are made of subsets, right? Therefore:   * The aggregation is when those subsets exists independently of the father set. As a monitor can be unplugged from the computer to be connected to another. * The composition is when those subsets depends of the existence of the father set. As a leaf is a part of a tree or liver is a part of a body.   These concepts talks about the kind of dependency between two objects or classes, conceptually. Directly in a program, in an aggregation, when the father object disposes, the aggregate objects should be disposed too. In the same scenario for a composition, composite son objects will persist then the father object dispenses. |
|  | add a comment |
| up vote**-2**down vote | How about this simple example:  An array of objects is a composition. An array of pointers to objects is an aggregation.  If I delete the first one, its contents vanish with it. The second one, on the other hand can vanish without affecting its members existence unless there is a specific method that deletes each object as its pointer is deleted. |
|  | add a comment |